**INSTITUTO POLITÉCNICO NACIONAL**

ESCUELA SUPERIOR DE CÓMPUTO


**NOMBRES:**

PADILLA MATIAS CRISTIAN MICHEL

SAUCILLO GONZÁLEZ JESSE OBED


**GRUPO:**

4BM1

**TRABAJO:**

Practica 1

"Agentes Reactivos"

**MATERIA:**

Fundamentos de Inteligencia Artificial

**FECHA**

19 de septiembre del 2023

INSTITUTO POLITÉCNICO NACIONAL

ESCOM

# PRACTICA 1. AGENTES REACTIVOS


## RESUMEN

En esta practica se realizo la creación de vehículos autónomos agentes partiendo de la modificación del código fuente “Agente” proporcionado por el profesor. A lo largo del presente trabajo se describe el proceso de desarrollo de agentes, sujeto a que parta de una madre nodriza, estos recojan muestras repartidas por un escenario rectangular de dimensiones n*m y su movimiento sea de tipo aleatorio.


## INTRODUCCIÓN

Los vehículos autónomos, también conocidos como "agentes autónomos", representan una evolución significativa en la forma en que concebimos y experimentamos la movilidad. En lugar de depender completamente de un conductor humano, estos vehículos están equipados con una variedad de sensores, software de inteligencia artificial y sistemas de control que les permiten tomar decisiones y realizar acciones de manera autónoma.

Para comprender mejor la teoría detrás de los vehículos agentes autónomos, es esencial explorar algunos de los conceptos clave y las tecnologías subyacentes que los hacen posibles.

Sensores y Percepción del Entorno: Los vehículos autónomos están equipados con una variedad de sensores, que incluyen cámaras, radares, lidar (detección y medición de la luz) y ultrasonidos. Estos sensores recopilan datos en tiempo real sobre el entorno circundante del vehículo. La percepción del entorno implica el procesamiento de estos datos para identificar objetos, como vehículos, y obstáculos. La visión por computadora y el aprendizaje automático permiten a los vehículos "ver" y comprender su entorno.

Toma de Decisiones y Planificación de Rutas: Una vez que se ha percibido el entorno, el vehículo autónomo debe tomar decisiones informadas. Esto implica la planificación de rutas, la toma de decisiones de conducción y la respuesta a situaciones imprevistas.

Control y Accionamiento: La ejecución de las decisiones se lleva a cabo a través del sistema de control del vehículo. Los sistemas de control avanzados permiten que el vehículo ajuste su velocidad y dirección en tiempo real para evitar obstáculos y seguir la ruta planificada de manera segura.

## DESARROLLO DE LA PRACTICA

El lenguaje que se utilizo para el desarrollo de esta practica fue Java. Para la creación de los vehículos agentes automáticos se modifico el proyecto “Agente” el cual contiene a su vez 4 clases: “Agentes”, “Agente”, “BackGroundPanel” y “Escenario”.

### Agentes

Esta clase representa a un agente autónomo en el entorno. Los agentes son regidos por las siguientes reglas:

```
if true then move randomly
if detect sample then pick up sample
if carrying samples & not at base then travel up gradient
if carrying samples & at mothership then drop samples
if detect obstacle then change direction
```

Es decir:

Los agentes interactúan con un entorno representado por una matriz bidimensional. bajo un movimiento aleatorio, movimiento creado con la función “MovimientoRandom”.

```
public void movimientoRandom(){
    Random random = new Random();

    int opcMovimiento = random.nextInt(4) + 1;

    switch (opcMovimiento){
        case 1: //Movimiento al norte
            movimientoNorte();
            break;
        case 2: //Moviendose al sur
            movimientoSur();
            break;
        case 3: //Movimiento al este
            movimientoEste();
            break;
        case 4: //Moviendose al oeste
            movimientoOeste();
            break;
        default:
            break;
    }
}
```

Los agentes se mueven en un tablero rectangular y recopilan muestras mientras navegan. Esto es posible gracias a la función “detectaSamples”.

```
//-----------Deteccion de samples-------------
public void detectaSamples(){
    if (matrix[i][j] == 2){
        contador++;
        System.out.println("\nCantidad de samples recogidos " + contador + "\n");
        matrix[i][j] = 0;
    }
}
```

Cada agente tiene sensores para detectar obstáculos y muestras en su entorno. Las funciones encargadas de esto son “obstaculoNorte”, “obstaculoSur”, “obstaculoEste” y “ObstaculoOeste”.

```
 //----------------Deteccion de obstaculos---
public boolean obstaculoNorte(){
     if (matrix[i][j-1] == 1) return true;
     return false;
}

public boolean obstaculoSur(){
     if (matrix[i][j+1] == 1) return true;
     return false;
}

public boolean obstaculoEste(){
    if (matrix[i+1][j] == 1) return true;
    return false;
}

public boolean obstaculoOeste(){
    if (matrix[i-1][j] == 1) return true;
    return false;
}
```

Pueden moverse en cuatro direcciones: norte, sur, este y oeste, pero su movimiento puede estar sujeto a cambios aleatorios o seguir una ruta específica hacia una ubicación objetivo. La función que describe este funcionamiento es “cambiaDireccion”.

```
public void cambiaDireccion(int direccionActual) {
    ArrayList<Integer> direccionesValidas = new ArrayList<>();

    // Comprobar direcciones válidas
    if (!limiteNorte() && !obstaculoNorte() && direccionActual != 1) {
        direccionesValidas.add(1); // Norte
    }
    if (!limiteSur() && !obstaculoSur() && direccionActual != 2) {
        direccionesValidas.add(2); // Sur
    }
    if (!limiteEste() && !obstaculoEste() && direccionActual != 3) {
        direccionesValidas.add(3); // Este
    }
    if (!limiteOeste() && !obstaculoOeste() && direccionActual != 4) {
        direccionesValidas.add(4); // Oeste
    }

    // Elegir una dirección aleatoria entre las válidas
    if (!direccionesValidas.isEmpty()) {
        Random random = new Random();
        int indiceAleatorio = random.nextInt(direccionesValidas.size());
        int nuevaDireccion = direccionesValidas.get(indiceAleatorio);

if (!direccionesValidas.isEmpty()) {
    Random random = new Random();
    int indiceAleatorio = random.nextInt(direccionesValidas.size());
    int nuevaDireccion = direccionesValidas.get(indiceAleatorio);

    // Mover en la nueva dirección
    switch (nuevaDireccion) {
        case 1:
            movimientoNorte();
            break;
        case 2:
            movimientoSur();
            break;
        case 3:
            movimientoEste();
            break;
        case 4:
            movimientoOeste();
            break;
    }
}
```

Adicionalmente, tienen métodos para pausar y reanudar su movimiento con las siguientes funciones.

```
public void relantizar(int num){
    try {
        sleep(num*100);
    } catch (InterruptedException ex) {
        ex.printStackTrace(System.out);
    }
}

public synchronized void pausar() {
    pausado = true;
}

public synchronized void reanudar() {
    pausado = false;
    notify();
    System.out.println("Hilo reanudado");
}
```

Los agentes originalmente contenían la representación de los personajes animados “Wall-e” y “Eva”, estos fueron cambiados por “Zombi1” y “Zombi2” del videojuego plantas contra zombis.

**Escenario**

Esta clase es la parte principal de la interfaz gráfica del programa. Crea un entorno rectangular donde los agentes se mueven y realizan sus acciones.

```
private void formaPlano()
{
    tablero = new JLabel[dim][dim];
    matrix = new int[dim][dim];

    int i, j;

    for(i=0;i<dim;i++)
        for(j=0;j<dim;j++)
        {
            matrix[i][j]=0;
            tablero[i][j]=new JLabel();
            tablero[i][j].setBounds(j*50+10,i*50+10,50,50);
            tablero[i][j].setBorder(BorderFactory.createDashedBorder(Color.white));
            tablero[i][j].setOpaque(false);
            this.add(tablero[i][j]);

            tablero[i][j].addMouseListener(new MouseAdapter() // Este listener nos
                {
                    @Override
                    public void mousePressed(MouseEvent e)
                    {
                           insertaObjeto(e);
                    }

                    @Override
                    public void mouseReleased(MouseEvent e)
                    {
                            insertaObjeto(e);
                    }
```

Administra la colocación de obstáculos, muestras y otros elementos en el entorno.

```
private void gestionaObstacle(ItemEvent eventObject)
{
    JRadioButtonMenuItem opt = (JRadioButtonMenuItem) eventObject.getSource();
    if(opt.isSelected())
       actualIcon = obstacleIcon;
    else actualIcon = null;
}

private void gestionaSample(ItemEvent eventObject)
{
    JRadioButtonMenuItem opt = (JRadioButtonMenuItem) eventObject.getSource();
    if(opt.isSelected())
       actualIcon = sampleIcon;
    else actualIcon = null;
}

private void gestionaMotherShip(ItemEvent eventObject)
{
    JRadioButtonMenuItem opt = (JRadioButtonMenuItem) eventObject.getSource();
    if(opt.isSelected())
       actualIcon = motherIcon;
    else actualIcon = null;
}
```

Controla la interacción entre los agentes y el entorno y proporciona opciones para configurar y ejecutar la simulación de los agentes autónomos.

```
private void gestionaRun(ActionEvent eventObject)
{
    if(!wallE.isAlive()) wallE.start();
    if(!eva.isAlive()) eva.start();
    settings.setEnabled(false);
    runButton.setEnabled(false);
    randomButton.setEnabled(false);

    enPausa = false;
    actualizaContadores();
}
```

También incluye opciones para pausar y reanudar la simulación.

```
private void gestionaPause(ActionEvent eventObject) {
    wallE.pausar();
    eva.pausar();

    enPausa = true;
    ((JButton) eventObject.getSource()).setText("Reanudar");
    settings.setEnabled(true);
    randomButton.setEnabled(true);
}

private void gestionaReanudar(ActionEvent eventObject) {
    wallE.reanudar();
    eva.reanudar();

    enPausa = false;
    ((JButton) eventObject.getSource()).setText("En Pause");
    settings.setEnabled(false);
}
```

Así como configurar la ubicación de la "mother ship" (nave madre), que es un elemento importante en la simulación.

```
public static ImageIcon getMotherIcon (){
    return motherIcon;
}
```

**Agente**

Esta clase es la entrada principal del programa y contiene el método "main". Su principal función es iniciar la interfaz gráfica del programa, que se manifiesta como un entorno donde los agentes se mueven y realizan sus acciones.

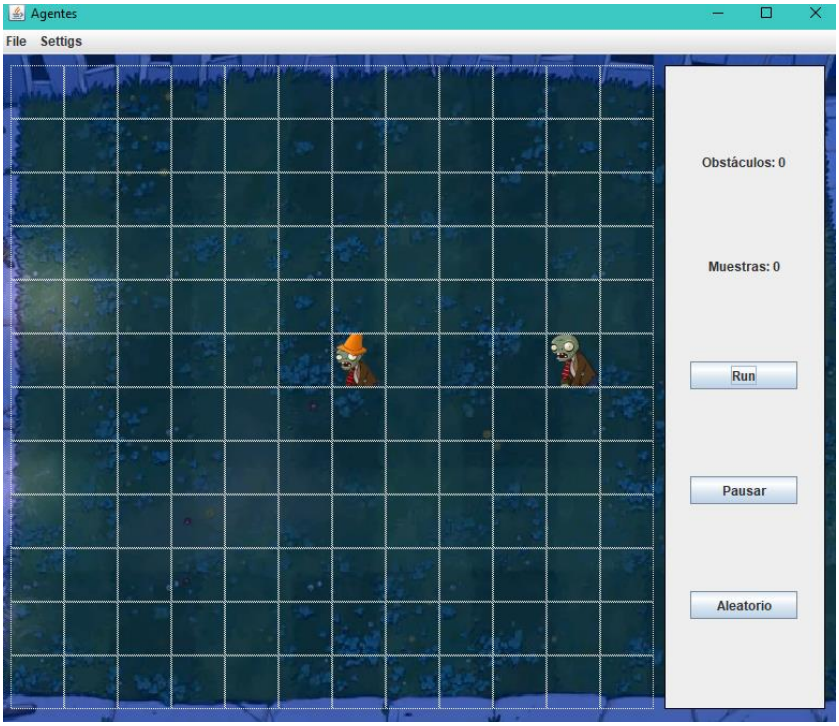
No se modifico nada de esta clase.

**BackGroundPanel**

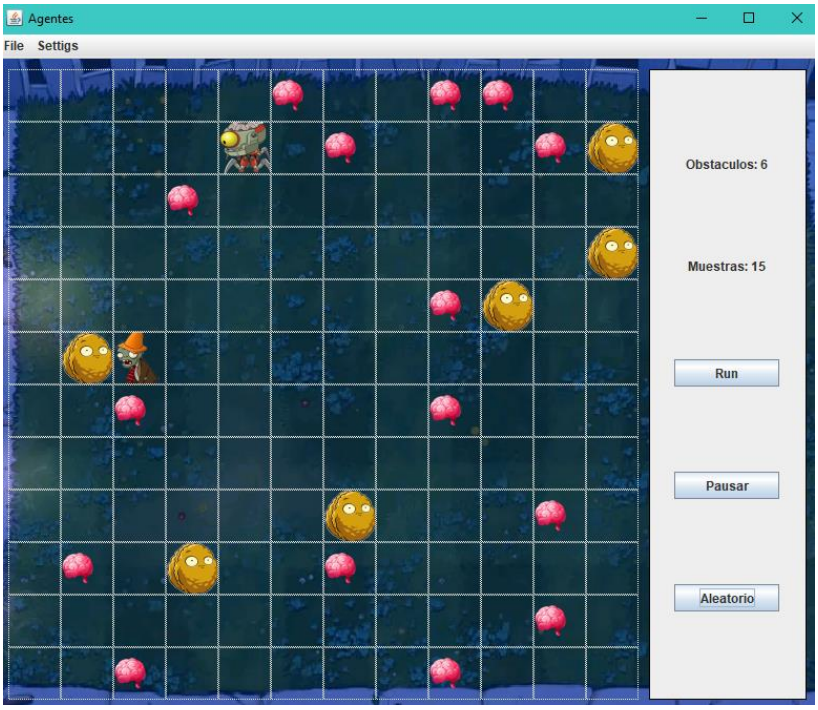
Esta clase es una extensión de JPanel y se utiliza para proporcionar un fondo de imagen en la interfaz gráfica. El fondo de imagen se utiliza como fondo para el entorno donde se mueven los agentes.

## DEMOSTRACIÓN DE FUNCIONAMIENTO

Ejecutando el programa:

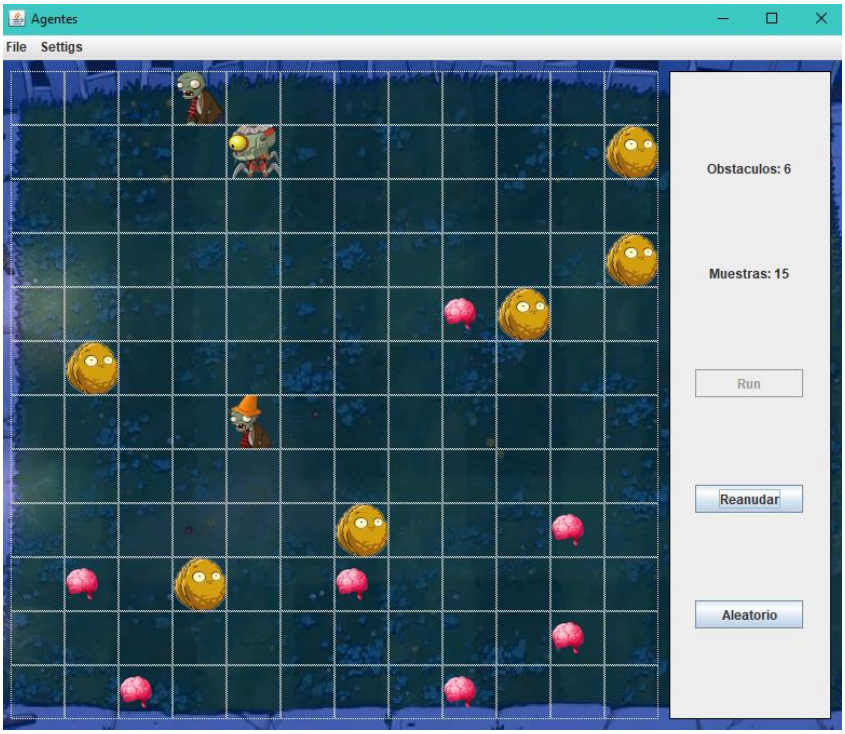



Colocando una configuración aleatoria.



Ejecutando el hilo de los agentes.


Pausando el hilo de los agentes.



**Liga de video de demostración:**

https://youtu.be/BwsvpeRDDjk

## CONCLUSIONES

A través de esta práctica, hemos explorado los conceptos clave detrás de los agentes autónomos, incluida la percepción del entorno mediante sensores, la toma de decisiones informadas y la planificación de rutas. Estos elementos son fundamentales para la autonomía de los vehículos autónomos y se aplican en una variedad de contextos, desde vehículos de transporte hasta robots industriales y sistemas de logística avanzados.

La flexibilidad y adaptabilidad de estos agentes son características esenciales, lo que les permite operar de manera eficiente en diversas situaciones y entornos cambiantes. Esta versatilidad los convierte en una tecnología prometedora para abordar desafíos en una amplia gama de aplicaciones, desde la seguridad vial hasta la automatización de procesos industriales complejos.

En un contexto más amplio, la práctica de agentes autónomos refleja el avance continuo de la inteligencia artificial y su capacidad para transformar la forma en que interactuamos con el mundo. A medida que esta tecnología evoluciona, es probable que veamos un aumento en la adopción de agentes autónomos en diversas industrias, lo que promete una mayor eficiencia, seguridad y comodidad en nuestras vidas cotidianas.

## BIBLIOGRAFÍA

*Aplicaciòn de agentes cognitivos en vehìculos autònomos inte*. (2018). Prezi.com. https://prezi.com/xch9b6zwftet/aplicacion-de-agentes-cognitivos-en-vehiculos-autonomos-inte/

Vidal, S. (2023, August 12). *¿Qué Son los Sistemas Autónomos? - TecnoBits ▷⃞→⃞*. Campus Habitat. https://tecnobits.com/que-son-los-sistemas-autonomos/

Tejal Sushir. (2023, September 15). *¿Qué son los agentes GPT y cómo funcionan?* Geekflare. https://geekflare.com/es/gpt-agents-explained/